An Arduino compatible CAN Bus architecture for sailing applications

Bruget Kevin, Benoit Clement, Olivier Reynet, Bernt Weber

An Arduino compatible CAN Bus architecture for sailing applications

Kévin Bruget, Benoît Clement, Olivier Reynet and Bernt Weber

Abstract This paper describes a Controller Area Network (CAN) Bus architecture based on Arduino compatible boards, to be used as an alternative communication system for robotic applications. This combines both, the robustness of CAN and the accessibility of Arduino software. The architecture is developed here to improve a Navigational Assistance System, which was initially created for disabled people. The system is composed of Arduino compatible boards, wired with various sensors and actuators, and communicating with an Human Machine Interface (HMI), directly accessible via a mobile phone or a tablet running on the open-source operating system Android. Information is transferred through the CAN bus architecture between multiple nodes (i.e. Arduino compatible boards) and the implementation of a CAN bootloader allows the reconfiguration of the nodes directly through the bus. The aim is to create a generic system able to work in various kinds of situations, adaptable to all kinds of users, including persons with all sorts of disabilities. This work will result in a demonstrator on a Miniji for the WRSC 2013 and an entirely joystick controlled boat for single handed sailing.

Kévin Bruget
ENSTA Bretagne, 2, Rue Francois Verny, 29200 Brest, France e-mail: kevin.bruget@ensta-bretagne.fr

Benoît Clement
ENSTA Bretagne Lab-STICC UMR CNRS 6285, 2, Rue Francois Verny, 29200 Brest, France e-mail: benoit.clement@ensta-bretagne.fr

Olivier Reynet
ENSTA Bretagne Lab-STICC UMR CNRS 6285, 2, Rue Francois Verny, 29200 Brest, France e-mail: oliver.reynet@ensta-bretagne.fr

Bernt Weber
Splashelec, 18, rue de Pont Louët, 29200 Brest, France e-mail: bernt.weber@splashelec.com

1 Introduction

Robotics in the sailing field is now a reality. Since 2008 with the first World Robotic Sailing Championship (WRSC) and International Robotic Sailing Conference (IRSC) [9, 10, 11, 31, 5], studies have shown that sailing robots can overcome the embryonic stage [18, 19, 23, 32]. Able to replace humans during competition or to realise autonomous measurements, its field of action becomes larger than before. Our approach is not to totally remove human action, but to assist it during information acquisition, decision process and execution (steering and trimming). Sailing globally has remained a field inaccessible for disabled people, because of the extreme mobility the sailor needs to acquire information and to manipulate commands of a boat. Sailing requires significant efforts that the disabled cannot afford. Splashelec aims to give disabled sailors access to those kinds of activities. The assistance system was initially composed of an electronic board and a joystick which allow a person to steer a boat manually as helmsman, forming part of a crew. When in need to free his hands, the helmsman can activate compass guided PID steering (autopilot).

Working with ENSTA Bretagne, an Android based Human Machine Interface (HMI) has been developed, in order to complete the system with visual navigational aid for disabled people [4]. To provide a more complete view of its environment for the skipper and to counterbalance his lack of mobility, some sensors, like wind sensor, compass and GPS were added to the system. Linked to an Arduino board, information is sent through Bluetooth to a tablet in charge of the information display.

At that point interfacing and cabling had already become complex and interfaces to connect extensions became a rare resource. Users then asked for joystick steering on smaller single handed boats, with need for supplementary actuators for the sails and more sensors to control the extra actuators. It was time to think about a new system architecture, with less connections for diminished cost and better reliability.

The present part of the project is the follow-up, which aims to develop a Controller Area Network (CAN [8]) bus interface board and the necessary software to allow communication using a more modular and flexible bus system, that is easily adaptable to all sorts of situations and disabilities. That is to say, developers can plug their own sensors to the system and show data directly through the tablet. On the software side, an Arduino bootloader compatible with CAN allows the programming of nodes directly through the bus. There is also a support for debugging messages relayed by the bus.

The main objective of this work is to provide to sailors, disabled or not, a navigational assistance system able to perform automated tasks (heeling limitation, autopilot, ...). It integrates the needed adaptability into the development process and aims to offer a solution for various kinds of disabilities. For example steering with a joystick compensates the lack of strength in the arm, the HMI centralises sensor information to compensate the lack of mobility and sensitivity (wind, boat speed,...). Furthermore, due to its open-source nature, the software system is entirely and easily modifiable and developers can add functionality or change the HMI according to users specific needs. Finally the system should be adaptable to every kind of boat.

This paper describes a complete system and its features, its architecture and the communication process between nodes. The demonstrator, an adapted Miniji boat will be presented at the WRSC 2013.

2 Related work

2.1 Low-level architecture of existing robot systems

Due to the complexity of tasks mobile robot are designed for, robots use generally embedded computers with substantial processing power and are often centered around one of them. Connecting to the robot's hardware is done using all sorts of interfaces available to the computer: USB, Ethernet, serial (RS232, RS435, RS485), and using hubs and port concentrators (e.g. NASA's [25] or Roboat [33]) where necessary. Employing microcontrollers in place of some port concentrators gives access to all the lower level electronic interfaces as I2C, SPI (NAO [29]).

In its simplest configuration, our system consists only of a joystick and a rudder actuator. In this configuration, there is no need for any embedded computer, the system implements only low-level reactive behaviours (compass controlled or joystick steering). A very simple microcontroller can execute these tasks, at low cost and with low current consumption.

As for the embedded computer centered architectures, everything must be wired to the one central microcontroller, but resources and available interfaces are more sparse on the lower abstraction level and the number of connected sensors and actuators is very limited. Centralized architectures become impractical when more components have to be connected to one simple microcontroller.

Bus systems offer a flexible and modular solution to interconnect microcontrollers, each having sensors and/or actuators attached. Historically used RS-485 multi-point serial cabling is more and more replaced with higher level CAN, with help of hardware implementations of the CAN protocol itself now integrated to microcontrollers. As for automotive and industrial field buses, CAN is today used in robots and results there in highly adaptable hardware architectures (Merten and Gross [24]).

Merten and Gross' robot architecture uses the CANopen [7] protocol. Open-source implementations of this protocol exist (CANFestival [6], CANopen SlaveLib [38] and CANopenNode [17]).

So we retained the CAN bus, but decided to go with an architecture able to work with very simple nodes and a network that is able to run without an embedded computer. Being able to run without embedded computer means being able to power-down the computer regularly for energy savings, or running entirely without it in simple configurations. And as we understand it, the three cited CANopen implementations do not allow to work without an embedded computer or high performance microcontroller. The missing piece seems to be a suitable simpler protocol, we are

not aware of a high-level (on-top of CAN) open-source protocol, simple enough for 8-bit microcontrollers, providing services necessary for sensors and actuators distributed over the bus, as for example synchronised sensor reading and distributed control loops.

2.2 Commercial marine electronics

2.2.1 Communication networks

Displays and autopilots installed on today's sailing yachts use mostly multi-drop serial buses with vendor specific proprietary protocols. Examples are Raymarine's SeaTalk [27] and NKE's Topline [13]. On some recent boats, CAN based, industry standard NMEA2000 [3] buses replace the proprietary protocols. For higher bandwidth requirements such as radar or echo sounder images, these buses are sometimes completed by extra Ethernet cabling (e.g. Furuno Navnet [14], Raymarine SeaTalkHS [26]).

Note that the Ethernet approach is power-hungry, costly and difficult to adapt to simple 8-bit microcontrollers, but industry choice CAN based NMEA2000 could be a good solution. The problem is that NMEA2000, and the J1939 protocol it is based on, are proprietary protocols [37]. Once again, as with CAN based robot architectures, CAN looks promising, but the missing piece is an adapted open source protocol, that would allow to design and integrate new hardware for different situations of handicap.

2.2.2 Autopilots

Commercial available autopilots are very close to joystick assisted steering for disabled people:

- Commercial autopilot actuators are used for steering by disabled. Splashelec uses for most boats actuators sold with commercial autopilots. A difference exists though in dimensioning for small vessels up to about 10 m length: a typical tiller autopilot uses an electrical motor with a power rating of about 10 to 20 W, which allows only for slow rudder movements compared to what a human helmsman can do. To give a disabled person using a joystick the same performance, the actuator has to be oversized compared to a commercial autopilot.
- Power electronics of commercial course computers as well Splashelec's model are designed around a H-bridge, allowing speed modulation for smooth rudder movements. There is no technical difference.
- Firmware of commercial autopilots is very specialized to course keeping following compass, wind or GPS data. All the autopilots known to us use PID closed loop control, calling the PID coefficients "rudder", "counter-rudder" and "autotrim" in the manuals.

Some commercial autopilots allow joysticks to control the rudder: pushing the joystick moves the rudder, when the joystick returns to the center, the rudder stays in position. A subset of these systems allows also for a proportional mode: the rudder follows the joystick (i.e. [28]). Such a system offers assistance to a disabled helmsman and some are used in this way [36].
Modifying firmware, which is not possible with commercially available autopilots, would allow for further adaptation to individual disabilities:

- joystick damping: many disabilities produce jerky hand movements. Simple position-averaging allows fine control of a sailing yacht.
- automatic adaptive correction of the rudder position corresponding to the joystick center: under sail, a boat is generally not completely balanced (i.e. weather helm), the rudder must keep an angle to steer straight. A simple proportional joystick must then stay continuously in an off-center position, against the spring, which is uncomfortable, and becomes impossible if the handicap implies reduced dexterity.

2.3 Existing assistance systems designed for disabled sailors

The company ”Hansa Sailing” [30] is a manufacturer selling a range of boats and assorted electric control systems destined for disabled sailors. Wiring applies star topology around a control box (Access Liberty Manual: [20]). Winch and rudder actuators have 2 wire connectors for DC-motors. The input devices need a 9 wire connection, typically a joystick with associated push buttons for mode selection. No sensors are associated to the DC-motors.

Steve Alvey’s company [22] sells electric controls for the Martin 16 and other sailing boat types used by disabled people [2]. He also designs specialized systems on purpose. An example is the boat steered by Hilary Lister around Britain; She uses a three straw sip-and-puff interface [16], and has access to a Raymarine autopilot.

The manual of the Martin 16 system [1] shows a Raymarine 3 pin socket labelled ”Sea Talk” for command interface connections (joystick, sip and puff, etc.), the bus system simplifying cabling. A second socket labelled ”Helm Drive” mates the standard connector of the used Raymarine ST4000/SPX-5 electric actuator (2 used pins for direct DC-motor connection). No additional sensors are added to the actuator.

Experience using our rudder only steering system showed us that using a position sensor enables tactile feedback, by establishing a relation between rudder and joystick position. And, in this setup, a bigger rudder angle makes water push harder on the rudder, as does the center return spring in the joystick. The helmsman feels the rudder, its position and the water force.

Our conclusion is that connection count of existing sailing assistance systems for disabled has reached a limit, where it becomes impractical and expensive, in the environment where every connector has to be waterproof and salt water resistant. Joystick box connections have of up to 9 wires, adding sensors to individual

actuators would complicate the cabling even more. Bus systems have already been employed to user-interfaces, we think that a bus system should be extended to the entire system, including the actuators and the associated sensors. This would simplify installations and allow more advanced interactions, better user experience and make the systems more modular, flexible and adaptable.

3 System description

The system is actually composed of two major parts, the Programmable Servo Controller (PSC) and the HMI, which displays sensor information and allows the skipper to activate the joystick or to use the autopilot. The communication between the bus and the HMI tablet uses Bluetooth, whereas internal communication is made entirely using the CAN protocol. Its behaviour can be assimilated to a closed-loop system (Fig. 1). Information which comes from the environment as any human input has influence on the actuators, that is to say the system will be able to perform automated tasks. An example is a standard course keeping autopilot, or joystick steering mode, with the autopilot taking over when necessary to limit heeling.

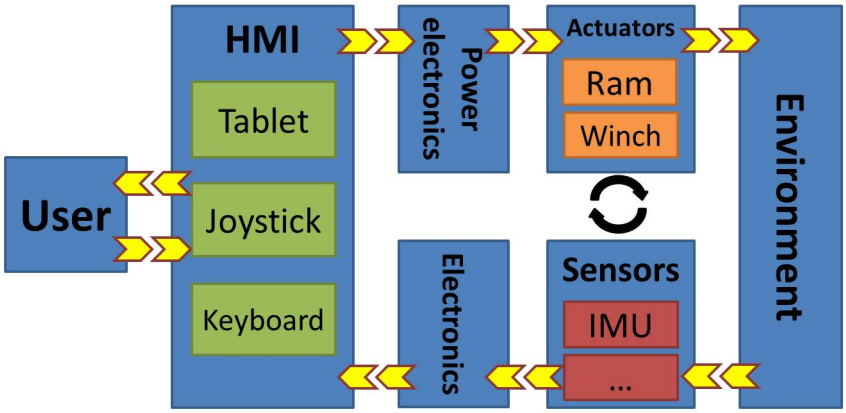

Fig. 1 The system is based on a closed-loop where the user can be removed to implement autonomous behaviour.

The programmable servo controller PSC (Fig. 2) is a key component of the system and can do the work of an autopilot course computer, which can steer to wind or compass when associated to the corresponding sensors. The PSC board (6 x 7.5 inches) contains a microcontroller, power electronics for an electric motor (ram or winch), an electric clutch and the power supply including filter circuits (see Fig. 3). Various interfaces for rudder angle sensor, joystick and control keyboard already exist and, in future versions, a CAN bus interface will be integrated directly to the

main circuit board. To add electric winches, we use one instance of the PSC for each.

Fig. 2 PSC of the Splashelec system in a waterproof case. It contains several inputs for the keyboard, the compass, the rudder angle sensor, the battery...

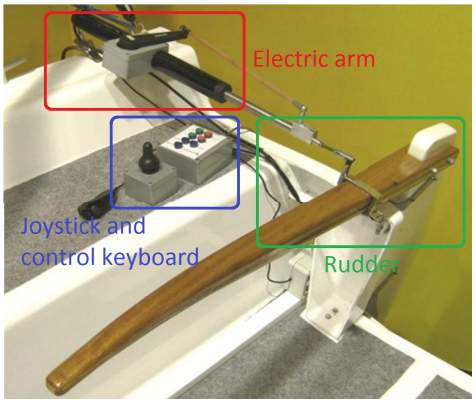

Fig. 3 The Splashelec system composed of an electric ram (in red), a joystick and a control keyboard (in blue) and the rudder (in green).

This PSC is based on open-source technology in order to allow the easy integration of new functionalities or to modify actual ones. The microcontroller is compatible to Arduino boards and can so be programmed with Arduino's Integrated Development Environment (IDE) software, which gives an access to the programming interface, existing libraries and various on-line examples. Results of this project (material and software) are publicly available on the Internet [34]. Wired to this box, multiple sensors such as an Inertial Measurement Unit (IMU), a wind sensor or loch-speedo are linked into a CAN architecture. When using more than one actuator, each one uses its own dedicated PSC, with instances of the power electronics,

local sensors, and a CAN bus interface. All the system, including mechanical parts, is transportable and can adapt to different boats.

The HMI, programmed for Android by using the Android Software Development Kit (SDK) tool for Eclipse [12], contains different areas (Fig. 4): in addition to textual information, some data, such as wind orientation and speed, compass and rudder angle, is also displayed in graphic form to allow a better understanding of the environment. A last area acts as a virtual keyboard to switch between autopilot and joystick mode and the user can modify the course to steer. The graphical layout of the HMI is realised in XML language and could be easily modified to fit needs of specific users. This results in a system (joystick, tablet display, extra keyboard,...), compatible with different disabilities, while still being usable by sailors without any disabilities.

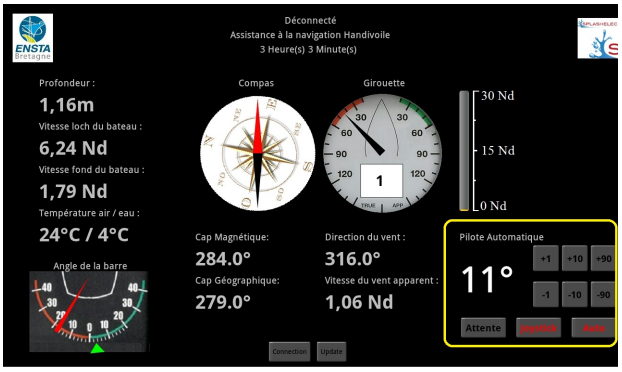

Fig. 4 The Human Machine Interface displays sensor information and contains a virtual keyboard (in yellow) to replace the physical one.

4 CAN Bus architecture

The need for a bus architecture came from the multi sensor and actuator problematic and the cabling and communication complexing with each added element. Used first in automotive applications, the CAN protocol is now widely available and starts migrating into many non-automotive applications. This open standardised high layer protocol provides a reliable message exchange system between various nodes.

The next section describes the protocol used in our system and its architecture. Finally the following section discusses the implementation of a Arduino CAN bootloader able to reconfigure nodes through the bus.

4.1 CAN protocol

The choice of the CAN protocol for our architecture arose from the need of a broadcast communication mechanism, that had to be easy to use, able to work with multiple nodes and which provides a reliable communication protocol to exchange information from sensors and actuators. Indeed CAN is able to detect errors with no less than three mechanisms: Cyclic Redundancy Check (CRC) to verify message integrity, Frame check to verify that data is sent in the correct shape and acknowledgments to guarantee reception. Besides, adding nodes to an existing CAN network can easily be done, which meets our needs for a modular architecture.

Adding new nodes to the network is straight forward: adding a new sensor with interfaces as for example NMEA 183 (National Marine Electronics Association) or I2C (Inter Integrated Circuit), implies reading the data by the connecting nodes microcontroller and to put the data in a CAN frame. Arduino allows to do this in very few lines of code, using libraries for CAN and a plethoric choice of sensor interfaces.

4.1.1 Higher-layer protocol

At this point appears the need of a higher level CAN protocol to organize data in the CAN frames. Various higher-layer CAN protocols already exist such as SAE J1939 used in NMEA 2000, CAN Kingdom, or CANopen but it appears that no one fits our need of a simple CAN protocol. Too complex or not adapted for 8-bit microcontrollers, CAN protocols are not widespread and their code is not always open-source. It has been decided to develop our own protocol based on our needs, called SimpleCAN, into an Arduino library containing the essential functions to support our architecture. The protocol is designed in a way permitting to add new features afterwards.

4.1.2 System architecture

Our architecture (Fig. 5) is based on Arduino compatibility and uses a CAN bus interface card we call the CANinterfacer. The CANinterfacer is compatible to an Arduino with a CAN shield on top. This on purpose designed board [35] uses an ATMEGA32U4 microcontroller as do the Arduino Leonardo and Micro. It is small in dimension (1.95 x 1.95 inch, slightly smaller than 5 x 5 cm) and can be programmed by USB using the Leonardo bootloader and the Arduino IDE. It can be powered from the CAN bus by an on-board switching power supply accepting from 7 to 32 Volts, most of the I/O pins are available for local connections.

In the system, a group of elements (actuators, sensors, HMI elements,...), wired to a CANinterfacer, becomes a CAN node (Fig. 6). That is to say, each CANinterfacer typically uses Arduino libraries to convert input from various sources, for example analog inputs, NMEA 183 or I2C connected sensors, into a CAN messages. Putting

a CANinterfacer between the new hardware and the bus to integrate it as standard node gives great flexibility.

In the same way, the upcoming version the PSC will contain an CANinterfacer and be a native node in our bus system. This type of node allows to integrate servo controlled actuators such as rams and winches with their associated sensors.

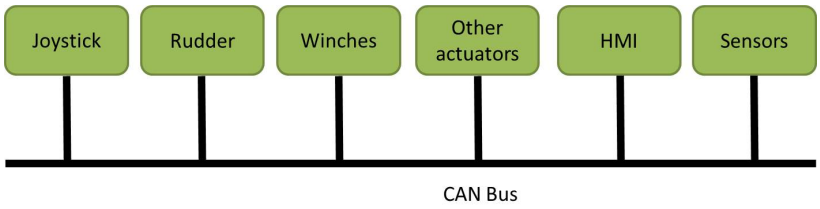

Fig. 5 CAN architecture of the system

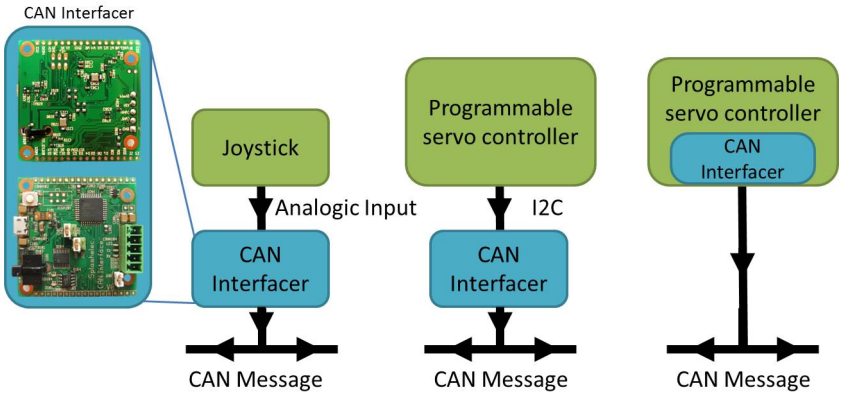

Fig. 6 Examples of a CAN node: The joystick and the Programmable Servo Controller (PSC) in its first version. The next version will have a CANinterfacer integrated into the board.

4.2 CAN Bootloader

The implementation of the CAN protocol opens the way to simplified programming of every node directly through the bus, one unique connection to the bus permits this. To obtain that, we need a new Arduino bootloader that accepts CAN programming commands for our CANinterfacer. Fabian Greif from the Robotics Club of Aachen,

Germany, has already worked on the subject with very close hardware [15] and built its own CAN bootloader, that allows updating firmware and local code from CAN messages. Small modifications have been made, in order suit connections of our CANinterfacer. [35].

The programming operation is initiated by a Python script that initiates the communication process between the PC connected programming node and the node which needs to be reconfigured. More concretely, the programming node will act as an In-System Programming (ISP) interface: it receives the new program by USB or serial port and sends it encapsulated in CAN messages to reprogram the specified node. The whole process is detailed in [35].

4.3 Demonstrator

The final objective of the current project is to build a demonstrator to show robotic functionality during WRSC 2013 [21] that proves the flexibility of such an architecture. The choice of the boat fell on a Miniji (Fig. 7) made available by the association "Handivoile Brest", which is based on a small scale replica of a historic America's Cup hull. It is a single-handed sailing boat, ordinarily steered by foot pedals or with a steering wheel. In a comfortable position, it offers vivid sensations to the sailor housed in a bucket seat. Inexpensive and very technical, the boat type was adopted by many French disabled sailors. But it is also used as sailing robot for the VAIMOS project of Institut Français de Recherche pour l'Exploration de la Mer (IFREMER) and ENSTA Bretagne [18, 19, 23].

Fig. 7 The system will be implemented on a Miniji sailboat to act as a demonstrator.

Furthermore, there is demand to increase the autonomy and safety, coming from instructors working around disabled sailing. The boat will be equipped with two electric winches and a rudder system, all interconnected by the CAN bus. By adding sensors to the bus (e.g GPS and IMU), this boat will be able to perform automated tasks, as does a sailing robot. Indeed, to increase security, we can limit the heeling

or restrict the navigational area. Instructors will also be able to take control of the boat with a remote controller for safety reasons or even to activate the autopilot if the skipper becomes unable to manipulate commands.

5 Conclusion

In conclusion we have a system based on a CAN bus architecture which allows developers to connect new sensors or actuators, and to reprogram the system easily using Arduino technology. Such a system assists the sailor during navigation and can automate complex tasks. It helps disabled by easing the access to the sailing activity, or gives any other people navigational assistance. Based on the open-source approach, electronics and software can be modified according to specific requirements, numerous opportunities for development exist. During WRSC 2013, a prototype will demonstrate robotic functionality. The final aim is to obtain products with new features derived from robotic sailing, encouraging people to develop their own system modifications.

References

1. Steve Alvey. *Martin 16 Power-Assist System - Mk IV, Operator Manual* http://www.martin16.com/uploads/auto_mkiv.pdf, 2005.
2. Steve Alvey. *Martin 16 Power-Assist System - Mk IV, Brochure* http://www.martin16.com/uploads/autobrochure.pdf, Cited 23 June 2013.
3. National Marine Electronics Association. *NMEA2000(TM) standard*, In: National Marine Electronics Association website http://www.nmea.org/content/nmea_standards/nmea_2000_ed3_00.asp, 2012.
4. N. Brocheton, K. Bruget, A. Wibaux, O. Reynet, B. Clement, and B. Weber. Systeme d'assistance a la navigation handivoile. In *Proceedings of Handicap 2012 : 7th congress on technical assistances for disabled people*, Paris, France, 2012.
5. J. Finnis C. Sauze. *Proceedings of the 5th International Robotic Sailing Conference*. Springer, 2012.
6. CanFestival. *CanFestival, a CANopen framework.* http://www.canfestival.org/, Cited 13 May 2013.
7. CiA. *CANopen*, In : CAN in Automation http://www.can-cia.org/index.php?id=systemdesign-canopen, 2013.
8. CiA. *CAN specifications*, In : CAN in Automation http://www.can-cia.org/index.php?id=can, Cited 13 May 2013.
9. Collective. *Proceedings of the 1st International Robotic Sailing Conference*. 2008.
10. Collective. *Proceedings of the 2nd International Robotic Sailing Conference*. 2009.
11. Collective. *Proceedings of the 3rd International Robotic Sailing Conference*. Universidade do Porto, 2010.
12. Eclipse. *Eclipse (software).* http://www.eclipse.org/, 2013.
13. Nke Marine Electronics. *nke marine electronics* http://www.nke-marine-electronics.com/home.html, Cited 23 June 2013.
14. Furuno. *Furuno Navnet* http://www.navnet.com/, 2012.

15. Fabian Greif. *CAN Bootloader*, In : Universal CAN library. Roboterclub Aachen e.V. http://www.kreatives-chaos.com/artikel/can-bootloader, 2010.
16. hilarylister.com. *Sip & Puff* http://www.hilarylister.com/d5483/hilary_s_boat/sip_amp_puff.aspx, 2012.
17. Janez. *CANopenNode* http://sourceforge.net/projects/canopennode/, Cited 23 June 2013.
18. L. Jaulin. Modelisation et commande dun bateau a voile. In *Proceedings of 3rd Conference Internationale Francophone dAutomatique*, Douz,Tunisie, 2004.
19. L. Jaulin, B. Clement, Y. Gallou, F. Le Bars, O. Menage, O. Reynet, and J. Sliwka. Suivi de route pour un robot voilier. In *Proceedings of 7th Conference Internationnale Francophone d'Automatique*, Grenoble, France, 2012.
20. Access Dinghy Sailing Systems Pty Ltd. *Access Dinghies OPERATIONS & SAFETY MANUAL - LIBERTY* http://www.sailingforall.com/imgs/74liberty_operation_&_safety_manual_pdf.pdf, Cited 25 June 2013.
21. F. Le Bars M. Melguen. *Official WRSC 2013 Website.* http://www.ensta-bretagne.eu/wrsc13/, 2013.
22. Martin16. *M16 sailboat for able and disabled sailors* http://www.martin16.com/, Cited 23 June 2013.
23. O. Menage, F. Gaillard, T. Gorgues, T. Terre, P. Rousseaux, S. Prigent, Y. Auffret, L. Dussud, B. Forest, M. Repecaud, L. Jaulin, B. Clement, Y. Gallou, and F. Le Bars. VAIMOS: Voilier autonome instrumente pour mesures oceanographiques de surface. In *Symposium on Vulnerability of coastal ecosystems to global change and extreme events*, Biarritz, France, 2011.
24. Matthias Merten and Horst-Michael Gross. Highly adaptable hardware architecture for scientific and industrial mobile robots. In *RAM*, pages 1130–1135. IEEE, 2008.
25. Eric Park, Linda Kobayashi, and Susan Y. Lee. Extensible hardware architecture for mobile robots. In *ICRA*, pages 3084–3089. IEEE, 2005.
26. Raymarine. *Ethernet*, In: Raymarine website http://www.raymarine.eu/view/?id=5537.
27. Raymarine. *SeaTalk/SeaTalk1* http://www.raymarine.eu/view/?id=5535&collectionid=26&col=5557, Cited 23 June 2013.
28. Raymarine. *Autopilot Joystick*, In: Raymarine website http://www.raymarine.com/view/?id=2705, Cited 24 June 2013.
29. Aldebaran Robotics. *NAO Software 1.14.3 documentation: low level architecture*, In: website of Aldebaran Robotics http://www.aldebaran-robotics.com/documentation/naoqi/sensors/dcm/low_level_architecture.html, Cited 23 June 2013.
30. Hansa Sailing. *Hansa Sailing* http://hansasailing.com/, Cited 23 June 2013.
31. A. Schlaefer and O. Blaurock. *Proceedings of the 4th International Robotic Sailing Conference*. Springer, 2011.
32. J. Sliwka, J. Nicola, R. Coquelin, F. Becket De Megille, B. Clement, and L. Jaulin. Sailing without wind sensor and other hardware and software innovations. In *Proceedings of the 4th International Robotic Sailing Conference (Springer Eds.)*, Lubeck, Germany, 2011.
33. Roland Stelzer and Karim Jafarmadar. The robotic sailing boat asv roboat as a maritime research platform. In *Proceedings of 22nd International HISWA Symposium*, 2012.
34. B. Weber. *Splashelec autopilot documentation*, In: The Splashelec Wiki http://wiki.splashelec.com/index.php/autopilot, Cited 13 May 2013.
35. B. Weber. *The CAN Interfacer*, In: The Splashelec Wiki. http://wiki.splashelec.com/index.php/caninterfacer, Cited 13 May 2013.
36. wetwheels.co.uk. *About the boat* http://www.wetwheels.co.uk/about-us/about-the-boat/, Cited 24 June 2013.
37. Wikipedia. *NMEA 2000*, In: English Wikipedia http://en.wikipedia.org/wiki/nmea_2000, Cited 23 June 2013.
38. Raphael Zulliger and Edouard TISSERANT. *CANopen SlaveLib* http://canopen.sourceforge.net/projects/slavelib/slavelib.html, Cited 23 June 2013.